# Conclusion

# Autonomous System Coding Challenge

Programmers must reason through system-wide interactions to generate codes for:

- monitoring
- hardware mode confirmation
- goal tracking
- detecting anomalies
- isolating faults
- diagnosing causes
- parameter estimation

- hardware reconfiguration
- fault recovery
- standby
- safing
- fault avoidance
- adaptive control
- control policy coordination

➡ poor reuse, poor coverage, error prone

# Solution: Part 1
# Model-based Programming

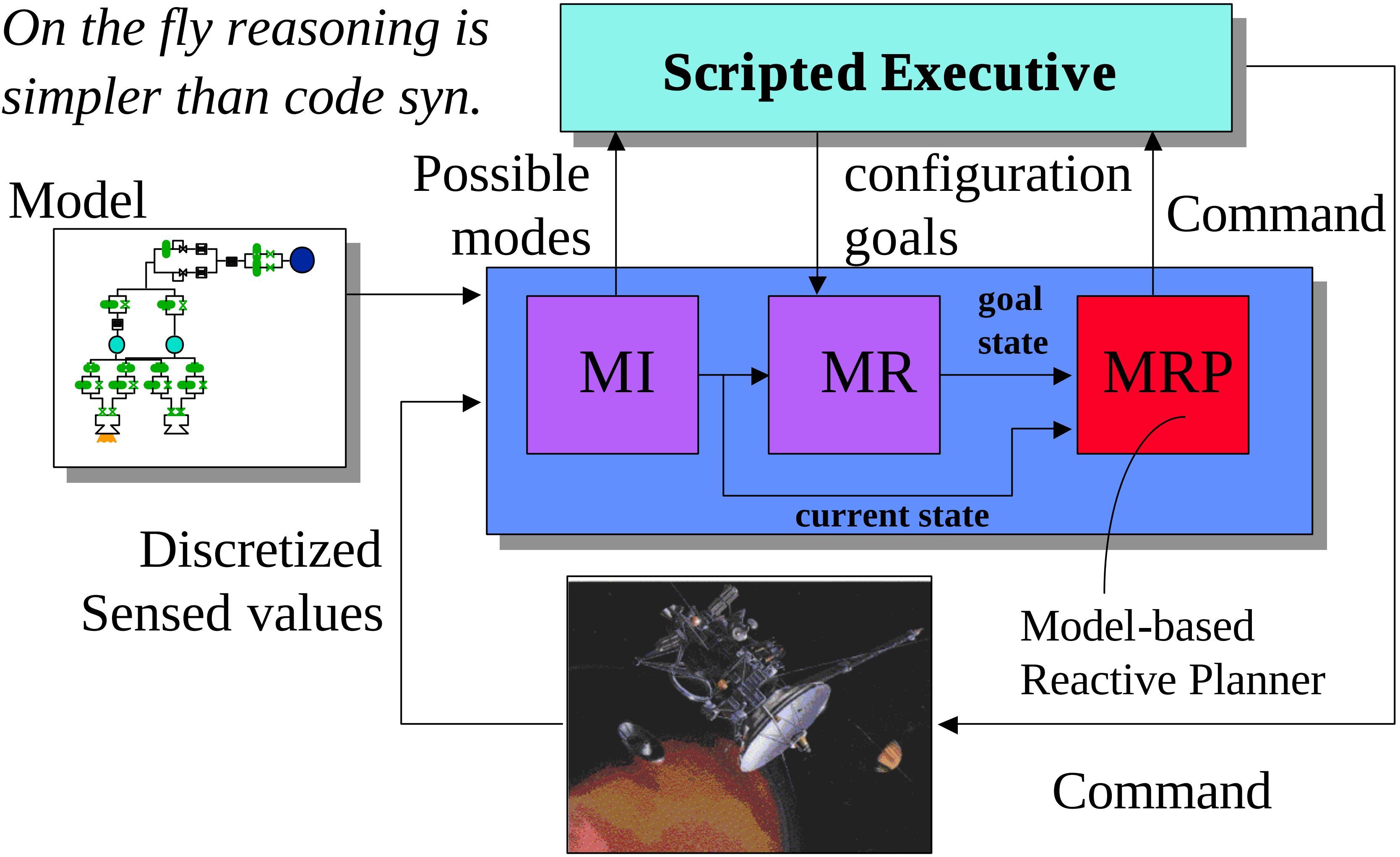
Programmers and operators generate breadth of functions from commonsense hardware models in light of mission-level goals.

Have engineers program in models, automate synthesis of code:

- models are compositional & highly reusable.
- generative approach covers broad set of behaviors.
- commonsense models are easy to articulate at concept stage and insensitive to design variations.

# Solution: Part 2
# Model-based Deductive Executive

*On the fly reasoning is simpler than code syn.*



Scripted Executive

Model

Possible modes

configuration goals

Command

MI

MR

goal state

MRP

current state

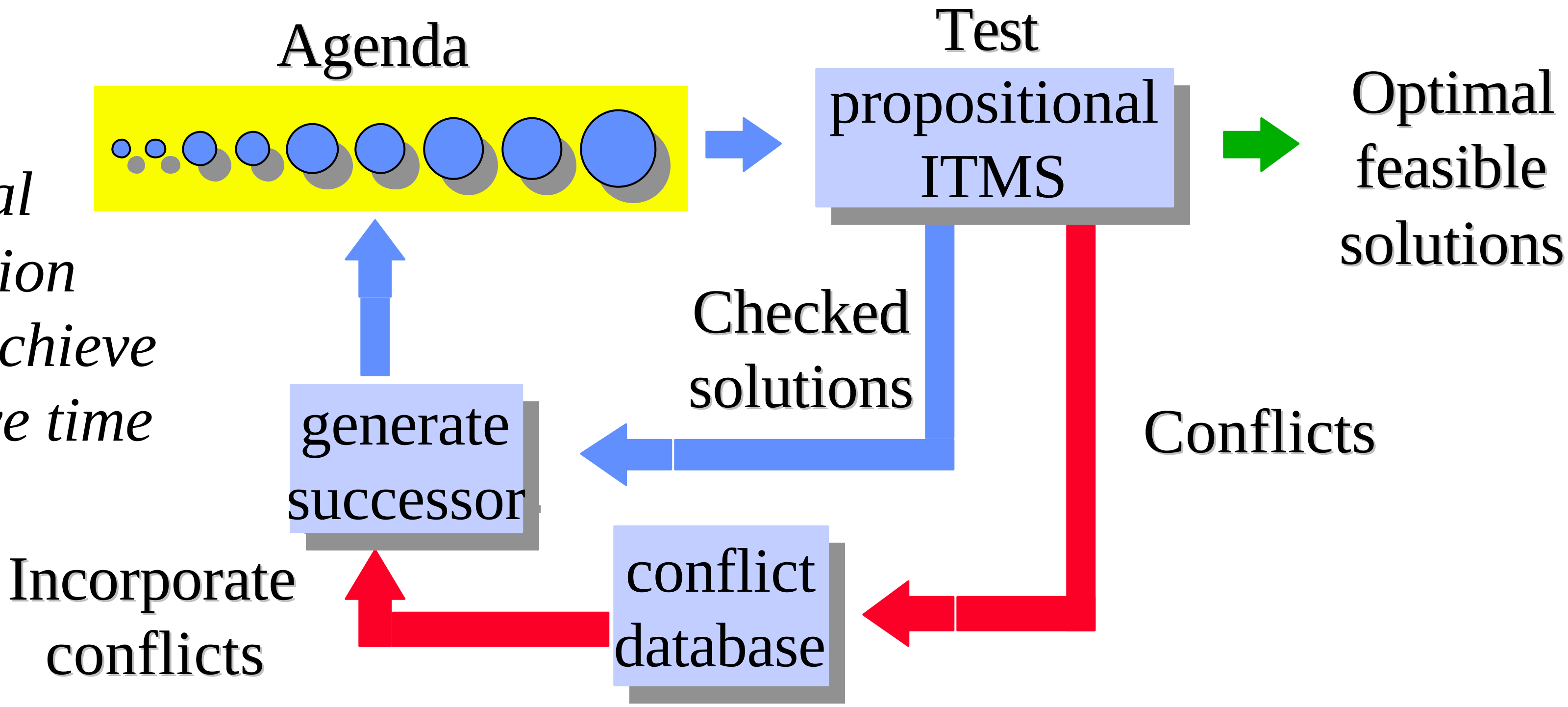
Discretized Sensed values

Model-based Reactive Planner

Command

# Solution: Part 3
# Risc-like Best-first, Deductive Kernel

*General deduction CAN achieve reactive time scales*



Agenda

Test propositional ITMS

Optimal feasible solutions

Checked solutions

Conflicts

generate successor

conflict database

Incorporate conflicts

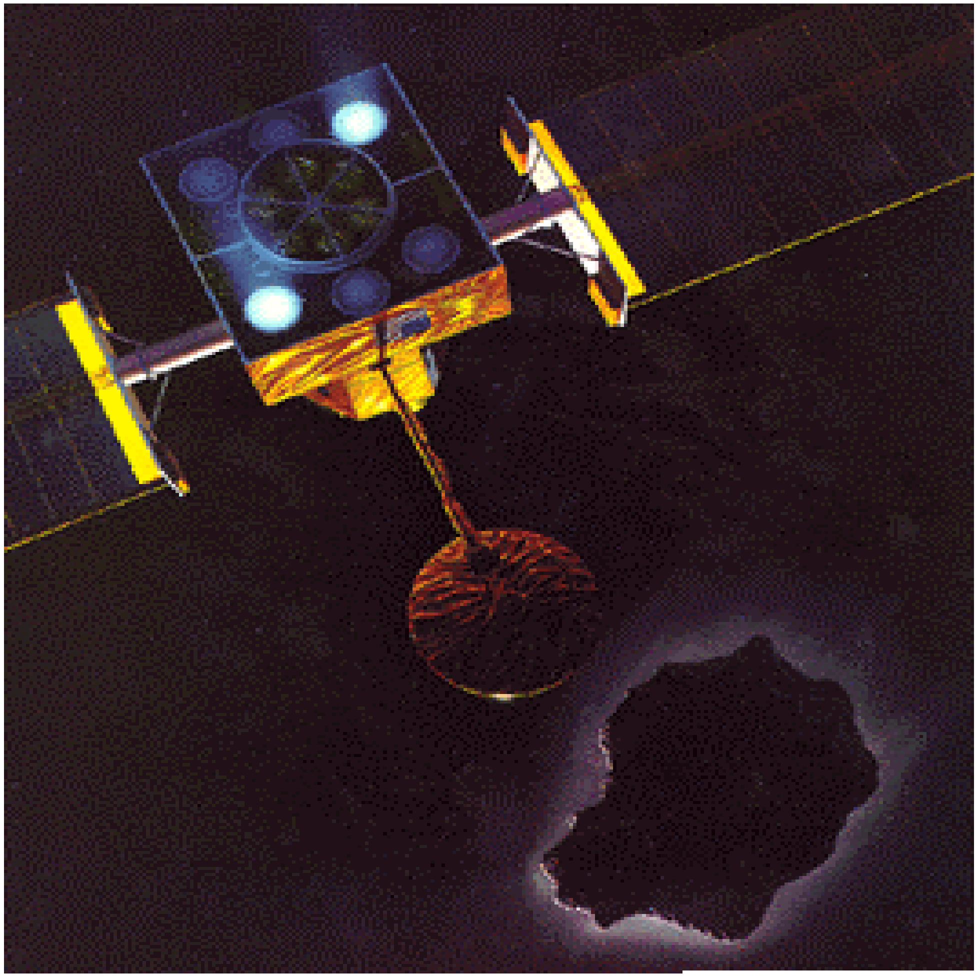
- Tasks and models compiled into propositional logic queries
- Conflicts dramatically focus search
- Careful enumeration grows agenda linearly
- ITMS efficiently tracks state changes in truth assignments

# Demonstration of Model-based Autonomy Capabilities

QuickTime™ and a
Photo - JPEG decompressor
are needed to see this picture

*Simulated mission of Saturn orbital insertion - Fall 1995*

*Flight demonstration on Deep Space One spacecraft Fall 1998*



# Future: Systems that Model & Adapt

QuickTime™ and a
Photo - JPEG decompressor
are needed to see this picture

- Spontaneous learning of failure dynamics.
- Stability analysis using qualitative phase portraits.
- Large-scale nonlinear adaptive code generation from models.
- Model-based learning starting from qualitative models alone.

# Future: Systems that Seek Information

- Instruments that design and execute elaborate experiments to model its environment and its own internal workings.
- Space probes that evaluate science ops and design missions.

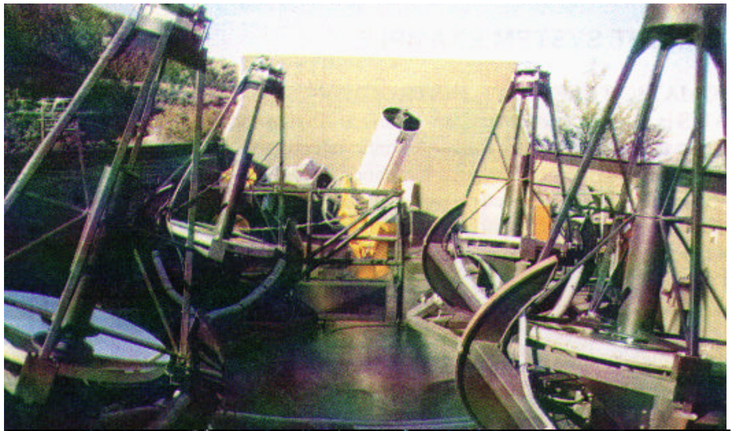
Bioreactor:
intelligent science instrument

# Future: Systems that Anticipate

QuickTime™ and a
Photo - JPEG decompressor
are needed to see this picture

- Predict critical failures for given context.
- Construct contingencies plans.
- Prepares backup resources to ensure fast response.

# Future: Systems that Collaborate



- Integrate heterogeneous reasoning schemes.
- Achieve coordination with out centralized control.
- Coordinate multiple systems to best achieve science objectives.